UNITED STATES LETTERS PATENT APPLICATION
FOR:

MULTI-MODAL METHOD FOR APPLICATION SWAPPING

INVENTOR:

William Patrick Tunney

Prepared by:

KENYON & KENYON
One Broadway
New York, NY 10004
(212) 425-7200

MULTI-MODAL METHOD FOR APPLICATION SWAPPING

BACKGROUND

5   **[001]**   Many Windows-based operating systems allow a user to swap between running applications by using the ALT+TAB key combination. This combination causes a display window to appear which includes graphical icons for all the currently running applications. By holding down the ALT key while repeatedly pressing the TAB key, a user may cycle through the graphical icons until she reaches the icon corresponding to

10   the running application she wishes to view. As the user cycles through the icons, each successive icon is highlighted. An exemplary highlight mark is a box-outline around the icon. Upon reaching the desired icon, the user may release the ALT+TAB key combination and the currently displayed application is swapped with a new application corresponding to the desired icon.

15

[002]   However, a concern with these systems occurs when the user has a large number of applications running at once. In this case, the user may have to press the TAB key numerous times before reaching the desired icon and swapping to the desired application. In one instance, the user may wish to access a reference window, e.g., an

20   API browser, that is running, but not accessed often. When the user does need to access the reference window, the user wishes to do so quickly. However, the user must press the TAB key numerous times to reach the reference window icon. In another instance, the user may wish to return to a recently viewed application. In some Windows-based operating systems, if the user previously minimized the application

25   windows using the WINDOWS_KEY+M key combination (e.g., in order to access data stored on the desktop) and then used the ALT+TAB key combination, the graphical icons of the most recently viewed applications appear last in the display window. Hence, the user is forced to tab through the entire list before reaching the desired icon. Additionally, if the user cycles through the list too quickly and passes over the desired

30   icon, the user must now awkwardly press the SHIFT key, in addition to holding down the ALT key and pressing the TAB key, in order to tab backward to the desired icon. Or the user must tab forward through the entire list again until reaching the desired icon.

These waste time and effort. As a result, the intended effectiveness of the ALT+TAB key combination is reduced.

[003]   As an alternative to the ALT+TAB generated display window, some systems provide a taskbar for listing the currently running applications. The taskbar is generally disposed across the bottom of the display screen and provides an application icon for each running application. The user typically uses the mouse to click on an application icon to view the corresponding application.

[004]   However, there is still a concern when the user has a large number of applications running at once. The width of the taskbar is limited to the width of the display screen. Thus, the width of each application icon is limited to an equally divisible portion of the taskbar. The more applications running, the greater the number of icons to fit on the taskbar and, hence, the narrower each icon width. So, where there are several applications running, the corresponding icons on the taskbar are too narrow for the user to read. The user instead must move the mouse over each icon as an information window pops up and displays an application descriptor. Or the user must click on the icon and display the application in order to identify it. Alternatively, the user may extend the height of the taskbar to provide multiple rows for the application icons. However, the more applications running, the greater the taskbar height and, hence, the less available display space on the screen. None of these options are ideal, all wasting time and effort.

[005]   Accordingly, there is a need in the art for a method to quickly and easily swap between currently running applications.

SUMMARY OF THE INVENTION

[006]   Embodiments of the present invention provide a method for swapping between currently running applications. These embodiments provide a multi-modal input mechanism that improves the effectiveness of the ALT+TAB key combination when application swapping, using a keyboard and a mouse. The method includes displaying the graphical icons corresponding to the currently running applications in response to an

ALT+TAB key combination. The method further includes selecting one of the graphical icons in response to a mouse input and displaying the running application corresponding to the selected icon. The method advantageously improves the speed and ease of application swapping when using the ALT+TAB key combination.

5

BRIEF DESCRIPTION OF DRAWINGS

[007]   FIG. 1 is an exemplary computer used for application swapping according to embodiments of the present invention.

10

[008]   FIG. 2 is a flowchart of an embodiment of a method according to the present invention.

[009]   FIG. 3 is a flowchart of an alternate embodiment of a method according
15   to the present invention.

[0010]   FIG. 4 is a flowchart of another alternate embodiment of a method according to the present invention.

20   ## DETAILED DESCRIPTION

[0011]   Embodiments of the present invention provide a multi-modal method for swapping between currently running applications. Embodiments use the ALT+TAB key combination and the mouse to select one of the applications to view. The ALT+TAB key combination causes a window to appear on the display screen, the window containing
25   graphical icons corresponding to the currently running applications. The mouse may then be used to click on one of the graphical icons, causing the application corresponding to the selected icon to appear on the display screen.

[0012]   By using the mouse with the ALT+TAB key combination, the user is able
30   to directly select the desired icon by clicking on it with the mouse rather than TAB to it. Accordingly, the multi-modal combination of the keyboard and mouse provides a quick

and easy way to swap between currently running applications, particularly when there is a large number of applications running at once.

[0013] FIG. 1 shows an exemplary computer that may be used to implement embodiments of the present invention. The computer 100 may include input/output devices, such as a keyboard 110, a mouse 120, and a display screen 130, and a processing core 140 to process inputs from the keyboard 110 and mouse 120 and outputs to the display screen 130. The core 140 may include, but is not limited to, a processor provided in communication with a system memory module, a storage device, and possibly other I/O devices (not shown). The processor may process the inputs and outputs. The memory may store program instructions to be executed by the processor and also may store variable data generated pursuant to program execution. The program instructions may represent an operating system of the computer 100 and/or application programs. In practice, the memory may be a memory system including one or more electrical, magnetic, or optical memory devices. The I/O devices may include serial and parallel ports for interface between the keyboard 110, mouse 120, display screen 130, and core 140.

[0014] Embodiments of the present invention may implement the multi-modal method by modifying an existing ALT+TAB key combination event handler to include routines to handle mouse-based events. Such events may include an event detecting a position of the mouse pointer relative to the graphical icons (a "mouse focus" event) and an event receiving a mouse click indicating selection of one of the icons (a "mouse click" event). When the event handler displays the graphical icons upon receipt of the ALT+TAB key combination, it may use information about the currently running applications including the icon of the application, a text string describing the icon, i.e., the application descriptor, the (x,y) coordinates and size of the icon on the display screen, an application icon id for each active application, and a pointer to the currently highlighted icon. The mouse-based events that are added to the existing event handler also make use of this information, as will be described below.

[0015] The mouse focus event may emulate the pressing of the TAB key. When the user moves the mouse pointer to the desired icon, the event handler may process the event in terms of the number of TAB key presses to be performed in order to move to and highlight the desired icon. To do this, the event handler may first detect the

5 (x,y) coordinates of the mouse pointer on the display screen. The event handler may then match the (x,y) coordinates of the mouse pointer with the region on the display screen occupied by a graphical icon. The application icon id of the matched graphical icon may then be retrieved.

10 **[0016]** If the application icon id of the currently highlighted icon and that of the icon currently pointed to are the same, then the event handler need not perform the mouse focus event. That is, the mouse pointer may be pointing to the already highlighted icon. So, the event handler need not swap applications. However, if not, then the event handler may calculate the number of TAB key presses required to move

15 from the currently highlighted icon to the icon currently pointed at. The event handler may then perform the number of TAB key presses to visually highlight the newly selected icon and to internally associate the pointer with the newly selected icon.

[0017] TABLE 1 shows exemplary code that may demonstrate a methodology for

20 the mouse focus event OnMouseMove that performs the functions as described above. This code is illustrative of what one might find in a Windows implementation of embodiments of the present invention.

```
25      // Mouse handler that monitors mouse coordinates and is called whenever
        // the mouse coordinates change. This extension to the ALT+TAB event handler
        // tests if the user is holding the ALT key while moving the mouse. If the user
        // is holding the ALT key and the mouse pointer is placed over an application
        // icon, OnKeyDown events are fired until the application icon is highlighted,
        // equating to TAB key presses.
30
        void OnMouseMove(UINT nFlags, CPoint point)
        {
                ApplicationLink *temp = head;
                int iconIndex = 0;
35              int count =0;
```

```
     // Cycle through all graphical icons in the list until either
     // the currently highlighted icon is found or the end of the
     // list is reached.
     while(temp != NULL)
5    {
             count++;
             if(temp->PointtInArea(point))
             {
                     iconIndex = count;
10                   break;
             }
             temp = temp->next;
     }

15   // if the mouse pointer points to an icon
     if(iconIndex != 0)
     {
             int difference = 0;

20           // If the pointed to icon is not the same as and is listed
             // after the currently highlighted icon, then calculate the
             // number of TABs to get to the pointed to icon.
             if(iconIndex > iconSelected)
             {
25                   difference = iconIndex-iconSelected;
             }
             // If the pointed to icon is the same as the currently
             // highlighted icon, then do nothing.
             else if (iconIndex == iconSelected)
30           {
                     return;
             }
             // If the pointed to icon is not the same as and is listed
             // before the currently highlighted icon, then calculate
35           // the number of TABs to get to the pointed to icon.
             else
             {
                     difference = (numberOfApplications-
                                     iconSelected)+iconIndex;
40           }

             // Fire the number of TABs to get to the pointed to icon
             for(int i=0; i<difference; i++)
             {
45                   OnKeyDown(81,0,0);
             }
     }
```

```
            OnMouseMove(nFlags, point);
    }
```

TABLE 1. Mouse Focus Event Source Code

5   **[0018]**   The mouse click event may emulate the release of the ALT key. After the user has reached the desired icon by using either the TAB key or the mouse pointer, the user may click a mouse button to select the desired icon. The event handler may then perform the ALT release function, which causes the application corresponding to the selected icon to appear on the display screen and simultaneously the window displaying

10 the graphical icons to disappear.   In these embodiments, either the left, right, or middle, if available, mouse button may be configured to trigger the mouse click event depending on the application.

[0019]   TABLE 2 shows exemplary code that may demonstrate a methodology for

15 the mouse click event OnLButtonDown that performs the functions as described above. This code is illustrative of what one might find in a Windows implementation of embodiments of the present invention.

```
            // Mouse handler that monitors when the left mouse button is clicked.
20          // If the mouse points to an application icon while the list of graphical icons
            // is displayed, then an event OnKeyUp is fired equating to the user releasing
            // the ALT key.

            void OnLButtonDown(UINT nFlags, CPoint point)
25          {
                    // if left mouse click & an icon is selected
                    if(iconSelected != 0)
                    {
                            ApplicationLink *temp = head;
30                          int iconIndex = 0;
                            int count =0;
```

```
                    // Cycle through all graphical icons in the list until either
                    // the currently highlighted icon is found or the end of the
                    // list is reached.
                    while(temp != NULL)
5                   {
                            count++;
                            if(temp->PointtInArea(point))
                            {
                                    iconIndex = count;
10                                  break;
                            }
                    temp = temp->next;
                    }

15                  // if the mouse pointer points to an icon
                    if(iconIndex != 0)
                            // display the application corresponding to the icon
                            OnKeyUp(17, 0, 0);
                    return;
20          }

            OnLButtonDown(nFlags, point);
        }
```

25   **TABLE 2**. Mouse Click Event Source Code

[0020] FIG. 2 is a flowchart of an embodiment of a method for application swapping according to the present invention. The processor may receive (205) an ALT+TAB key combination input from the keyboard 110. The processor may then run

30 the event handler to handle the ALT+TAB event, where the processor may display (210) graphical icons corresponding to the currently running applications on the display screen 130.

[0021] While the user continues to hold the ALT key down, the processor may

35 wait to receive the mouse-based events. The processor may receive (215) a mouse focus event from the mouse 120 as the user moves the mouse pointer to point to the desired graphical icon. On the display screen 130, a graphical icon may be highlighted (220) as the mouse pointer moves over it. Concurrently, the processor may resolve the coordinates of the pointer with the graphical icon and associate the pointer with the

40 graphical icon, as previously described.

[0022] After the desired icon is highlighted, if the user clicks on it, the processor may receive (225) a mouse click event. The processor may then release the display of graphical icons and switch (230) to the application corresponding to the selected icon, emulating a release of the ALT key. Anytime thereafter, the user may physically release the ALT key.

[0023] If the user has not clicked on an icon, then the processor may receive (215) additional mouse focus events, if any, while waiting to receive (225) a mouse click event. After the user clicks on an icon with the mouse, the processor may switch (230) to the application corresponding to the currently highlighted icon.

[0024] FIG. 3 is a flowchart of an alternate embodiment of a method for application swapping according to the present invention. The processor may continue to run (360) other operating system programs until it receives (305) an ALT+TAB key combination as input. After receiving the ALT+TAB key combination input, the processor may then display (310) graphical icons corresponding to all the currently running applications. In the display window, the icon of the application currently selected may be highlighted (315). The user may continue to hold down the ALT key.

[0025] If the processor receives (320) a TAB key press as input, the next icon in the display window may be highlighted (325). If the processor does not receive (320) a TAB key input, then the processor may determine whether the user has moved the mouse pointer to point to an icon. If so, the processor may receive (330) the mouse pointer input and the mouse focus event may be triggered. The processor may then resolve the mouse pointer input using a technique as previously described, for example, and highlight (335) the icon pointed to.

[0026] Next, the processor may determine whether the user has clicked on the highlighted icon with the mouse. If so, the processor may receive (340) the mouse click input and the mouse click event may be triggered. Upon receiving the mouse click input, the processor may perform the routine that releases (345) the ALT key. The

processor may then switch (355) to the application corresponding to the highlighted icon. Anytime thereafter, the user may physically release the ALT key.

[0027] If the processor does not receive (330, 340) a mouse pointer input or a mouse click input, then the processor may determine (350) whether the user has released the ALT key. If so, the processor may switch (355) to the application corresponding to the highlighted icon. If the user has not released the ALT key, then the processor may wait for receipt of a TAB key input (320), a mouse pointer input (330), or a mouse click input (340) in order to select and switch to one of the running applications.

[0028] After the ALT+TAB+mouse combination has completed, the processor may then run (360) other operating system programs until receiving the next ALT+TAB input.

[0029] FIG. 4 is a flowchart of another alternate embodiment of a method for application swapping according to the present invention. In this embodiment, the user need not continue to hold down the ALT key while using the TAB key or the mouse to select an application to switch to. Instead, the user may press the ALT+TAB key combination and then release it. The processor may receive (405) the ALT+TAB input and display (410) graphical icons corresponding to the currently running applications on the display screen 130. The icon corresponding to the currently selected application may be highlighted (415). The processor may continue to display the icons for a predetermined time period after the user has released the ALT key. A timer may start (420) after the icons are displayed and the ALT key released in order to measure the predetermined time period.

[0030] While the timer has not expired or the user has not pressed the ESC key (425), the processor may determine whether input from either the keyboard 110 or the mouse 120 has been received. Similar to FIG. 3, if the processor receives (430) a TAB input, then the next icon in the display window may be highlighted (435). If the processor receives (440) a mouse pointer input, then the processor may highlight (445)

the icon that the pointer points to. If the processor receives (450) a mouse click input, then the processor may switch (455) to the application corresponding to the highlighted icon.

5   **[0031]**   After the timer has expired or the user has pressed an ESC key (425), the processor may end the ALT+TAB+mouse combination and then run (460) other operating system programs until receiving the next ALT+TAB input.

[0032]   It is to be understood that the present application is not limited to the
10   ALT+TAB key combination, but may include any key strokes, individually or in combination, that may be configured to perform embodiments of the present invention.

It is to be further understood that embodiments are not limited to the combination of a keyboard and a mouse. Any other device, such as a trackball, a touch pad, a touch
15   screen, a stylus, or a laser pen, capable of providing input to a processor may be used to replace either or both the keyboard and/or mouse.

[0033]   Embodiments of the present invention may be implemented using any type of computer, such as a general-purpose microprocessor, programmed according to
20   the teachings of the embodiments. The embodiments of the present invention thus also includes a machine readable medium, which may include instructions used to program a processor to perform a method according to the embodiments of the present invention. This medium may include, but is not limited to, any type of disk including floppy disk, optical disk, and CD-ROMs.
25

[0034]   It may be understood that the structure of the software used to implement the embodiments of the invention may take any desired form, such as a single or multiple programs. It may be further understood that the method of an embodiment of the present invention may be implemented by software, hardware, or a
30   combination thereof.

[0035]   The above is a detailed discussion of the preferred embodiments of the invention. The full scope of the invention to which applicants are entitled is defined by the claims hereinafter. It is intended that the scope of the claims may cover other
35   embodiments than those described above and their equivalents.